Our Ref. No. 042390.P9654
Express Mail No.: EL466332021US

UNITED STATES PATENT APPLICATION

FOR

CONTROLLING ACCESS TO MULTIPLE MEMORY ZONES IN AN ISOLATED EXECUTION ENVIRONMENT

INVENTORS:

Carl M. Ellison
Roger A. Golliver
Howard C. Herbert
Derrick C. Lin
Francis X. McKeen
Gil Neiger
Ken Reneris
James A. Sutton
Shreekant S. Thakkar
Mittal Millind

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Blvd., 7th Floor
Los Angeles, CA 90025-1026
(714) 557-3800

BACKGROUND

1. Related Application

This application claims the benefit U.S. Provisional Patent Application No. 60/198,227 filed on March 31, 2000.

2. Field of the Invention

This invention relates to microprocessors. In particular, the invention relates to processor security.

3. Description of Related Art

Advances in microprocessor and communication technologies have opened up many opportunities for applications that go beyond the traditional ways of doing business. Electronic commerce (E-commerce) and business-to-business (B2B) transactions are now becoming popular and are conducted throughout global markets at continuously increasing rates. Unfortunately, while modern microprocessor systems provide users convenient and efficient methods of doing business, communicating and transacting, they are also vulnerable for unscrupulous attacks. Examples of these attacks include virus, intrusion, security breach, and tampering, to name a few. Computer security, therefore, is becoming more and more important to protect the integrity of the computer systems and increase the trust of users.

Threats caused by unscrupulous attacks may be in a number of forms. An invasive remote-launched attack by hackers may disrupt the normal operation of a system connected to thousands or even millions of users. A virus program may corrupt code and/or data of a single-user platform.

Existing techniques to protect against attacks have a number of drawbacks. Anti-virus programs can only scan and detect known viruses. Security co-processors or smart cards using cryptographic or other security techniques have limitations in speed performance, memory capacity, and flexibility. Further, redesigning operating systems creates software compatibility issues and requires tremendous investment in development efforts.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1A is a diagram illustrating an operating system according to one embodiment of the invention.

Figure 1B is a diagram illustrating the accessibility of various elements in the operating system and the processor according to one embodiment of the invention.

Figure 1C is a diagram, similar to Figure 1B, illustrating the accessibility of various elements in the operating system and the processor, in which, an isolated memory area includes a plurality of memory zones each of which is associated with a subsystem, respectively, according to one embodiment of the invention.

Figure 1D is a diagram illustrating a computer system in which one embodiment of the invention can be practiced.

Figure 2A is a diagram illustrating an embodiment of the isolated execution circuit shown in Figure 1D according to one embodiment of the invention.

Figure 2B is a diagram illustrating an access manager shown in Figure 2A according to one embodiment of the invention.

Figure 3 is a diagram illustrating a multi-memory zone access checking circuit according to one embodiment of the invention.

Figure 4 is a flowchart illustrating a process to control access to multi-memory zones of the isolated memory area for isolated execution according to one embodiment of the invention.

DESCRIPTION

The present invention is a method, apparatus and system to control memory accesses to multiple memory zones in an isolated execution environment. A processor having a normal execution mode and an isolated execution mode generates an access transaction. The access transaction is configured using a configuration storage that stores configuration settings. The configuration settings include a plurality of subsystem memory range settings defining multiple memory zones. The access transaction also includes access information. A multi-memory zone access checking circuit, coupled to the configuration storage, checks the access transaction using at least one of the configuration settings and the access information. The multi-memory zone access checking circuit generates an access grant signal if the access transaction is valid.

In one embodiment, the configuration settings include an isolated setting, a plurality of subsystem memory range settings, and an execution mode word. The access information includes a physical address. The configuration storage includes a setting storage to store the isolated setting that defines an isolated memory area in a memory external to the processor. The isolated setting includes a base register, a mask register and a length register to store a base value, a mask value and a length value, respectively. A combination of at least two of the base, mask and length values can be used to form the isolated setting. The configuration storage further includes a subsystem memory range storage that stores the plurality of subsystem memory range settings. Each subsystem memory range setting defines a memory zone for a subsystem in the isolated memory area. Each subsystem memory range setting includes a subsystem mask value, a subsystem base value and a subsystem length value. A combination of at least two of the subsystem base, mask and length values can be used to define a memory zone. The subsystem memory range storage further includes ID values for each subsystem to identify each subsystem and the subsystem's associated memory zones as defined by the subsystem memory range setting. The configuration storage also includes a processor control register to store the execution mode word. The execution mode word is asserted as an execution mode word signal identifying a currently active subsystem and indicating that the processor is configured in the isolated execution mode. The isolated memory area is only accessible to the processor when it is configured in the isolated execution mode.

In one embodiment, the multi-memory zone access checking circuit includes a subsystem address detector. The subsystem address detector detects if the physical address is within a currently active subsystem's associated memory zone as defined by the subsystem memory range setting for the subsystem. If the physical address is within the subsystem's associated memory zone, the subsystem address detector generates a subsystem address matching signal. An access grant generator is coupled to the subsystem address detector and the processor control register. The access grant generator generates an access grant signal if both the subsystem address matching signal and the execution mode word signal, identifying the currently active subsystem operating in the isolated execution mode, are asserted. Thus, only if the physical address requested by a processor, operating in an isolated execution mode, is within the correct memory zone for the currently initialized subsystem will the access transaction be granted.

In the following description, for purposes of explanation, numerous details are set forth in order to provide a thorough understanding of the present invention. However, it will be appreciated to one skilled in the art that these specific details are not required in order to practice the present invention. In other instances, well-known electrical structures and circuits are shown in block diagram form in order not to obscure the present invention.

ARCHITECTURE OVERVIEW

One principle for providing security in a computer system or platform is the concept of an isolated execution architecture. The isolated execution architecture includes logical and physical definitions of hardware and software components that interact directly or indirectly with an operating system of the computer system or platform. An operating system and the processor may have several levels of hierarchy, referred to as rings, corresponding to various operational modes. A ring is a logical division of hardware and software components that are designed to perform dedicated tasks within the operating system. The division is typically based on the degree or level of privilege, namely, the ability to make changes to the platform. For example, a ring-0 is the innermost ring, being at the highest level of the hierarchy. Ring-0 encompasses the most critical, privileged components. In addition, modules in Ring-0 can also access to lesser privileged data, but not vice versa. Ring-3 is the outermost ring, being at the lowest level of the hierarchy. Ring-3 typically encompasses users or applications level and has the least privilege protection. Ring-1 and ring-2 represent the intermediate rings with decreasing levels of privilege.

Figure 1A is a diagram illustrating a logical operating architecture 50 according to one embodiment of the invention. The logical operating architecture 50 is an abstraction of the components of an operating system and the processor. The logical operating architecture 50 includes ring-0 10, ring-1 20, ring-2 30, ring-3 40, and a processor nub loader 52. The processor nub loader 52 is an instance of an processor executive (PE) handler. The PE handler is used to handle and/or manage a processor executive (PE) as will be discussed later. The logical operating architecture 50 has two modes of operation: normal execution mode and isolated execution mode. Each ring in the logical operating architecture 50 can operate in both modes. The processor nub loader 52 operates only in the isolated execution mode.

Ring-0 10 includes two portions: a normal execution Ring-0 11 and an isolated execution Ring-0 15. The normal execution Ring-0 11 includes software modules that are critical for the operating system, usually referred to as kernel. These software modules include primary operating system (e.g., kernel) 12, software drivers 13, and hardware drivers 14. The isolated execution Ring-0 15 includes an operating system (OS) nub 16 and a processor nub 18. The OS nub 16 and the processor nub 18 are instances of an OS executive (OSE) and processor executive

(PE), respectively. The OSE and the PE are part of executive entities that operate in a protected environment associated with the isolated area 70 and the isolated execution mode. The processor nub loader 52 is a protected bootstrap loader code held within a chipset in the system and is responsible for loading the processor nub 18 from the processor or chipset into an isolated area as will be explained later.

Similarly, ring-1 20, ring-2 30, and ring-3 40 include normal execution ring-1 21, ring-2 31, ring-3 41, and isolated execution ring-1 25, ring-2 35, and ring-3 45, respectively. In particular, normal execution ring-3 includes N applications 42_1 to 42_N and isolated execution ring-3 includes K applets 46_1 to 46_K.

One concept of the isolated execution architecture is the creation of an isolated region in the system memory, referred to as an isolated area, which is protected by both the processor and chipset in the computer system. The isolated region may also be in cache memory, protected by a translation look aside (TLB) access check. Access to this isolated region is permitted only from a front side bus (FSB) of the processor, using special bus (e.g., memory read and write) cycles, referred to as isolated read and write cycles. The special bus cycles are also used for snooping. The isolated read and write cycles are issued by the processor executing in an isolated execution mode. The isolated execution mode is initialized using a privileged instruction in the processor, combined with the processor nub loader 52. The processor nub loader 52 verifies and loads a ring-0 nub software module (e.g., processor nub 18) into the isolated area. The processor nub 18 provides hardware-related services for the isolated execution.

One task of the processor nub 18 is to verify and load the ring-0 OS nub 16 into the isolated area, and to generate the root of a key hierarchy unique to a combination of the platform, the processor nub 18, and the operating system nub 16. The processor nub 18 provides the initial set-up and low-level management of the isolated area including verification, loading, and logging of the operating system nub 16, and the management of a symmetric key used to protect the operating system nub's secrets. The processor nub 18 may also provide application programming interface (API) abstractions to low-level security services provided by other hardware.

The operating system nub 16 provides links to services in the primary OS 12 (e.g., the unprotected segments of the operating system), provides page management within the isolated area, and has the responsibility for loading ring-3 application modules 45, including applets 46_1 to 46_K, into protected pages allocated in the isolated area. The operating system nub 16 may also load ring-0 supporting modules.

The operating system nub 16 may choose to support paging of data between the isolated area and ordinary (e.g., non-isolated) memory. If so, then the operating system nub 16 is also responsible for encrypting and hashing the isolated area pages before evicting the page to the ordinary memory, and for checking the page contents upon restoration of the page. The isolated mode applets 46_1 to 46_K and their data are tamper-resistant and monitor-resistant from all software attacks from other applets, as well as from non-isolated-space applications (e.g., 42_1 to 42_N), dynamic link libraries (DLLs), drivers and even the primary operating system 12. Only the processor nub 18 or the operating system nub 16 can interfere with or monitor the applet's execution.

Figure 1B is a diagram illustrating accessibility of various elements in the operating system 10 and the processor according to one embodiment of the invention. For illustration purposes, only elements of ring-0 10 and ring-3 40 are shown. The various elements in the logical operating architecture 50 access an accessible physical memory 60 according to their ring hierarchy and the execution mode.

The accessible physical memory 60 includes an isolated area 70 and a non-isolated area 80. The isolated area 70 includes applet pages 72 and nub pages 74. The non-isolated area 80 includes application pages 82 and operating system pages 84. The isolated area 70 is accessible only to elements of the operating system and processor operating in isolated execution mode. The non-isolated area 80 is accessible to all elements of the ring-0 operating system and processor.

The normal execution ring-0 11 including the primary OS 12, the software drivers 13, and the hardware drivers 14, can access both the OS pages 84 and the application pages 82. The normal execution ring-3, including applications 42_1 to 42_N, can access only to the application pages 82. Both the normal execution ring-0 11 and ring-3 41, however, cannot access the isolated area 70.

The isolated execution ring-0 15, including the OS nub 16 and the processor nub 18, can access both the isolated area 70, including the applet pages 72 and the nub pages 74, and the non-isolated area 80, including the application pages 82 and the OS pages 84. The isolated execution ring-3 45, including applets 46_1 to 46_K, can access only to the application pages 82 and the applet pages 72. The applets 46_1 to 46_K reside in the isolated area 70.

Figure 1C is a diagram, similar to Figure 1B, illustrating the accessibility of various elements in the operating system and the processor, in which, an isolated memory area 70 includes a plurality of memory zones (75_0-75_N) each of which is associated with a subsystem, respectively, according to one embodiment of the invention. For illustration purposes, only elements of ring-0 10 and ring-3 40 are shown. The various elements in the logical operating architecture 50 access an accessible physical memory 60 according to their ring hierarchy and the execution mode. The accessible physical memory 60 includes an isolated area 70 and a non-isolated area 80. In this embodiment shown in Figure 1C, the isolated memory area 70 is divided up into a plurality of memory zones (75_0-75_N) each of which is associated with a subsystem. Particularly, there is a master memory zone 75_0 associated with a master subsystem 71 and a plurality of other memory zones (75_1-75_n) each of which is associated with an operating system to define a subsystem. This allows the present invention the ability to support concurrent multiple subsystems allowing for increased platform functionality.

To support the multiple subsystems, one of the subsystems is defined as the master subsystem 71. As illustrated in Figure 1C, for exemplary purposes, the master subsystem 71 resides in master memory zone-0 75_0 and includes the processor nub 18 (processor executive (PE)) as processor nub pages 74. The processor nub 18 is loaded into one portion of the master memory zone-0 75_0. The master subsystem 71 is configured to include the entire isolated memory area 70 allowing it to control the memory used by the other zones. All other subsystems are configured as subsets of the master subsystem 70 and exclude the memory used by the processor nub 18. The master subsystem 71 is delegated with managing and creating the other subsystems and the other multiple memory zones (75_1-75_N). Particularly, the processor nub code executing in the master subsystem 71 provides the management and maintenance functions for the other subsystems such as allocating, configuring, initializing, and cleaning up the other memory zones. The master subsystem 71 is the only subsystem that can create additional subsystems by defining memory zones (e.g. 75_1-75_N) in the isolated memory area 70. Also, only code executing from the master subsystem may change the configuration of the other subsystems. Further, no subsystem can examine or modify the contents of the master subsystem 71 or any other subsystem.

As shown in Figure 1C, the present invention can have N number of memory zones to support N number of subsystems operating with N number of operating systems (OS's) and their corresponding OS nubs 16. Accordingly, the isolated memory area 70 can include multiple memory zones (75_1-75_N) each of which is associated with an operating system to define a subsystem. Only one memory zone and operating system is active at a time. Each OS nub corresponds to either the primary OS 12 or one of the multiple OS's 17, respectively. For example, a first subsystem could reside in memory zone-1 75_1 and includes the OS nub-1 pages 76_1 and applet pages 72_1. As an example, the OS nub-1 pages 76_1 could correspond to the OS nub for the primary OS 12 which could be, for example, a Windows™ operating system. Further, a second subsystem may reside in memory zone-2 75_2 and includes one of the multiple OS nubs 16 as OS nub-2 pages 76_2 and applet pages 72_2. As an example, a Linux™ operating system could be a second subsystem residing in memory zone-2 75_2. Furthermore, other operating systems can be located in the other remaining memory zones 75_3-75_N.

The non-isolated area 80 includes application pages 82 and operating system pages 84. The isolated area 70 is accessible only to elements of the operating system that is currently associated with the initialized subsystem and when the processor is operating in the isolated execution mode. The non-isolated area 80 is accessible to all elements of the ring-0 operating system and processor.

The normal execution ring-0 11 includes the primary OS 12, other multiple OS's 17, the software drivers 13, and the hardware drivers 14, all of which can access both the OS pages 84 and the application pages 82. The normal execution ring-3, including applications 421 to 42N, can access only the application pages 82. Both the normal execution ring-0 11 and ring-3 41, however, cannot access the isolated area 70.

The isolated execution ring-0 15 includes the OS nubs 16 and the processor nub 18. Only the currently active subsystem associated with a particular memory zone (75_1-75_N) in the isolated memory area 70, as previously discussed, can access that memory zone and its associated OS Nub pages (76_1-76_N) and applet pages (72_1-72_N). Further, no subsystem can examine or modify the contents of the master subsystem 71 or any other subsystem. However, the isolated execution ring-0 15, including the OS nubs 16 and the processor nub 18, can access the non-isolated area 80, including the application pages 82 and the OS pages 84. Also, the isolated execution ring-3 45, including applets 46_1 to 46_K, can access the application pages 82.

Figure 1D is a diagram illustrating a computer system 100 in which one embodiment of the invention can be practiced. The computer system 100 includes a processor 110, a host bus

120, a memory controller hub (MCH) 130, a system memory 140, an input/output controller hub (ICH) 150, a non-volatile memory, or system flash, 160, a mass storage device 170, input/output devices 175, a token bus 180, a motherboard (MB) token 182, a reader 184, and a token 186. The MCH 130 may be integrated into a chipset that integrates multiple functionalities such as the isolated execution mode, host-to-peripheral bus interface, memory control. Similarly, the ICH 150 may also be integrated into a chipset together or separate from the MCH 130 to perform I/O functions. For clarity, not all the peripheral buses are shown. It is contemplated that the system 100 may also include peripheral buses such as Peripheral Component Interconnect (PCI), accelerated graphics port (AGP), Industry Standard Architecture (ISA) bus, and Universal Serial Bus (USB), etc.

The processor 110 represents a central processing unit of any type of architecture, such as complex instruction set computers (CISC), reduced instruction set computers (RISC), very long instruction word (VLIW), or hybrid architecture. In one embodiment, the processor 110 is compatible with an Intel Architecture (IA) processor, such as the Pentium® processor series, the IA-32™ and the IA-64™. The processor 110 includes a normal execution mode 112 and an isolated execution circuit 115. The normal execution mode 112 is the mode in which the processor 110 operates in a non-protected environment, or a normal environment without the security features provided by the isolated execution mode. The isolated execution circuit 115 provides a mechanism to allow the processor 110 to operate in an isolated execution mode. The isolated execution circuit 115 provides hardware and software support for the isolated execution mode. This support includes configuration for isolated execution, definition of an isolated area, definition (e.g., decoding and execution) of isolated instructions, generation of isolated access bus cycles, and generation of isolated mode interrupts.

In one embodiment, the computer system 100 can be a single processor system, such as a desktop computer, which has only one main central processing unit, e.g. processor 110. In other embodiments, the computer system 100 can include multiple processors, e.g. processors 110, 110a, 110b, etc., as shown in Figure 1D. Thus, the computer system 100 can be a multi-processor computer system having any number of processors. For example, the multi-processor computer system 100 can operate as part of a server or workstation environment. The basic description and operation of processor 110 will be discussed in detail below. It will be appreciated by those skilled in the art that the basic description and operation of processor 110 applies to the other processors 110a and 110b, shown in Figure 1D, as well as any number of other processors that may be utilized in the multi-processor computer system 100 according to one embodiment of the present invention.

The processor 110 may also have multiple logical processors. A logical processor, sometimes referred to as a thread, is a functional unit within a physical processor having an architectural state and physical resources allocated according to some partitioning policy. Within the context of the present invention, the terms "thread" and "logical processor" are used to mean the same thing. A multi-threaded processor is a processor having multiple threads or multiple logical processors. A multi-processor system (e.g., the system comprising the processors 110, 110a, and 110b) may have multiple multi-threaded processors.

The host bus 120 provides interface signals to allow the processor 110 or processors 110, 110a, and 110b to communicate with other processors or devices, e.g., the MCH 130. In addition to normal mode, the host bus 120 provides an isolated access bus mode with corresponding interface signals for memory read and write cycles when the processor 110 is configured in the isolated execution mode. The isolated access bus mode is asserted on memory accesses initiated while the processor 110 is in the isolated execution mode. The isolated access bus mode is also asserted on instruction pre-fetch and cache write-back cycles if the address is within the isolated area address range and the processor 110 is initialized in the isolated execution mode. The processor 110 responds to snoop cycles to a cached address within the isolated area address range if the isolated access bus cycle is asserted and the processor 110 is initialized into the isolated execution mode.

The MCH 130 provides control and configuration of memory and input/output devices such as the system memory 140 and the ICH 150. The MCH 130 provides interface circuits to recognize and service isolated access assertions on memory reference bus cycles, including isolated memory read and write cycles. In addition, the MCH 130 has memory range registers (e.g., base and length registers) to represent the isolated area in the system memory 140. Once configured, the MCH 130 aborts any access to the isolated area that does not have the isolated access bus mode asserted.

The system memory 140 stores system code and data. The system memory 140 is typically implemented with dynamic random access memory (DRAM) or static random access memory (SRAM). The system memory 140 includes the accessible physical memory 60 (shown in Figure 1B). The accessible physical memory includes a loaded operating system 142, the isolated area 70 (shown in Figure 1B), and an isolated control and status space 148. The loaded operating system 142 is the portion of the operating system that is loaded into the system memory 140. The loaded OS 142 is typically loaded from a mass storage device via some boot code in a boot storage such as a boot read only memory (ROM). The isolated area 70, as shown in Figure 1B, is the memory area that is defined by the processor 110 when operating in the isolated execution mode. Access to the isolated area 70 is restricted and is enforced by the processor 110 and/or the MCH 130 or other chipset that integrates the isolated area functionalities. The isolated control and status space 148 is an input/output (I/O)-like, independent address space defined by the processor 110 and/or the MCH 130. The isolated control and status space 148 contains mainly the isolated execution control and status registers. The isolated control and status space 148 does not overlap any existing address space and is accessed using the isolated bus cycles. The system memory 140 may also include other programs or data which are not shown.

The ICH 150 represents a known single point in the system having the isolated execution functionality. For clarity, only one ICH 150 is shown. The system 100 may have many ICH's similar to the ICH 150. When there are multiple ICH's, a designated ICH is selected to control the isolated area configuration and status. In one embodiment, this selection is performed by an external strapping pin. As is known by one skilled in the art, other methods of selecting can be used, including using programmable configuring registers. The ICH 150 has a number of functionalities that are designed to support the isolated execution mode in addition to the traditional I/O functions. In particular, the ICH 150 includes an isolated bus cycle interface 152, the processor nub loader 52 (shown in Figure 1A), a digest memory 154, a cryptographic key storage 155, an isolated execution logical processing manager 156, and a token bus interface 159.

The isolated bus cycle interface 152 includes circuitry to interface to the isolated bus cycle signals to recognize and service isolated bus cycles, such as the isolated read and write bus cycles. The processor nub loader 52, as shown in Figure 1A, includes a processor nub loader code and its digest (e.g., hash) value. The processor nub loader 52 is invoked by execution of an appropriate isolated instruction (e.g., Iso-Init) and is transferred to the isolated area 70. From the isolated area 80, the processor nub loader 52 copies the processor nub 18 from the system flash (e.g., the processor nub code 18 in non-volatile memory 160) into the isolated area 70, verifies and logs its integrity, and manages a symmetric key used to protect the processor nub's secrets. In one embodiment, the processor nub loader 52 is implemented in read only memory (ROM). For security purposes, the processor nub loader 52 is unchanging, tamper-resistant and non-substitutable. The digest memory 154, typically implemented in RAM, stores the digest (e.g., hash) values of the loaded processor nub 18, the operating system nub 16, and any other critical modules (e.g., ring-0 modules) loaded into the isolated execution space. The cryptographic key storage 155 holds a symmetric encryption/decryption key that is unique for the platform of the system 100. In one embodiment, the cryptographic key storage 155 includes internal fuses that are programmed at manufacturing. Alternatively, the cryptographic key storage 155 may also be created with a random number generator and a strap of a pin. The isolated execution logical processing manager 156 manages the operation of logical processors operating in isolated execution mode. In one embodiment, the isolated execution logical processing manager 156 includes a logical processor count register that tracks the number of logical processors participating in the isolated execution mode. The token bus interface 159 interfaces to the token bus 180. A combination of the processor nub loader digest, the processor nub digest, the operating system nub digest, and optionally additional digests, represents the overall isolated execution digest, referred to as an isolated digest. The isolated digest is a fingerprint identifying the ring-0 code controlling the isolated execution configuration and operation. The isolated digest is used to attest or prove the state of the current isolated execution.

The non-volatile memory 160 stores non-volatile information. Typically, the non-volatile memory 160 is implemented in flash memory. The non-volatile memory 160 includes the processor nub 18.

The processor nub 18 provides the initial set-up and low-level management of the isolated area 70 (in the system memory 140), including verification, loading, and logging of the operating system nub 16, and the management of the symmetric key used to protect the operating system nub's secrets. The processor nub 18 may also provide application programming interface (API) abstractions to low-level security services provided by other hardware. The processor nub 18 may also be distributed by the original equipment manufacturer (OEM) or operating system vendor (OSV) via a boot disk.

The mass storage device 170 stores archive information such as code (e.g., processor nub 18), programs, files, data, applications (e.g., applications 42_1 to 42_N), applets (e.g., applets 46_1 to 46_K) and operating systems. The mass storage device 170 may include compact disk (CD) ROM 172, floppy diskettes 174, and hard disk 176, and any other magnetic or optic storage devices. The mass storage device 170 provides a mechanism to read machine-readable media.

I/O devices 175 may include any I/O devices to perform I/O functions. Examples of I/O devices 175 include controller for input devices (e.g., keyboard, mouse, trackball, pointing device), media card (e.g., audio, video, graphics), network card, and any other peripheral controllers.

The token bus 180 provides an interface between the ICH 150 and various tokens in the system. A token is a device that performs dedicated input/output functions with security functionalities. A token has characteristics similar to a smart card, including at least one reserved-purpose public/private key pair and the ability to sign data with the private key. Examples of tokens connected to the token bus 180 include a motherboard token 182, a token reader 184, and other portable tokens 186 (e.g., smart card). The token bus interface 159 in the ICH 150 connects through the token bus 180 to the ICH 150 and ensures that when commanded to prove the state of the isolated execution, the corresponding token (e.g., the motherboard token 182, the token 186) signs only valid isolated digest information. For purposes of security, the token should be connected to the digest memory.

When implemented in software, the elements of the present invention are the code segments to perform the necessary tasks. The program or code segments can be stored in a machine readable medium, such as a processor readable medium, or transmitted by a computer data signal embodied in a carrier wave, or a signal modulated by a carrier, over a transmission medium. The "processor readable medium" may include any medium that can store or transfer information. Examples of the processor readable medium include an electronic circuit, a semiconductor memory device, a ROM, a flash memory, an erasable programmable ROM (EPROM), a floppy diskette, a compact disk CD-ROM, an optical disk, a hard disk, a fiber optic medium, a radio frequency (RF) link, etc. The computer data signal may include any signal that can propagate over a transmission medium such as electronic network channels, optical fibers, air, electromagnetic, RF links, etc. The code segments may be downloaded via computer networks such as the Internet, Intranet, etc.

CONTROLLING ACCESS TO MULTIPLE MEMORY ZONES IN AN ISOLATED EXECUTION ENVIRONMENT

Figure 2A is a diagram illustrating an embodiment of the isolated execution circuit 115 shown in Figure 1D according to one embodiment of the invention. The isolated execution circuit 115 includes a core execution circuit 205 and an access manager 220.

The core execution circuit 205 includes an instruction decoder and execution unit 210 and a translation lookaside buffer (TLB) 218. The instruction decoder and execution unit 210 receives an instruction stream 215 from an instruction fetch unit. The instruction stream 215 includes a number of instructions. The instruction decoder and execution unit 210 decodes the instructions and executes the decoded instructions. These instructions may be at the micro- or macro- level. The instruction decoder and execution unit 210 may be a physical circuit or an abstraction of a process of decoding and execution of instructions. In addition, the instructions may include isolated instructions and non-isolated instructions. The instruction decoder and execution unit 210 generates a virtual address 212 when there is an access transaction, caused by the execution of instructions. The TLB 218 translates the virtual address 212 into a physical address.

The core execution circuit 205 interfaces with the access manager 220 via control/status information 222, operand 224, and access information 226. The control/status information 222 includes control bits to manipulate various elements in the access manager 220 and status data from the access manager 220. The operand 224 includes data to be written to and read from the access manager 220. The access information 226 includes address (e.g., the physical address provided by the TLB 218), read/write, and access type information.

The access manager 220 receives and provides the control/status information 222 and the operand 224, and receives the access information 226 from the core execution circuit 205 as a result of instruction execution, and further receives an external isolated access signal 278 from another processor in the system. The external isolated access signal 278 is asserted when another processor in the system attempts to access the isolated memory area. The access manager 220 generates a processor isolated access signal 272 and an access grant signal 274. The processor isolated access signal 272 may be used to generate an isolated bus cycle sent to devices (e.g., chipsets) external to the processor 110 to indicate that the processor 110 is executing an isolated mode instruction. The processor isolated access signal 272 and the access grant signal 274 may also be used internally by the processor 110 to control and monitor other isolated or non-isolated activities.

Figure 2B is a diagram illustrating the access manager 220 shown in Figure 2A according to one embodiment of the invention. The access manager 220 includes a configuration storage 250 and a multi-memory zone access checking circuit 270. The access manager 220 exchanges the operand 224 with and receives the access information 226 from the core execution circuit 205 shown in Figure 2A. The access manager 220 also receives the external isolated access signal 278 from another processor as previously shown in Figure 2A. The access information 226 includes a physical address 282, a read/write (RD/WR#) signal 284 and an access type 286. The access information 226 is generated during an access transaction by the processor 110. The access type 286 indicates a type of access, including a memory reference, an input/output (I/O) reference, and a logical processor access. The logical processor access includes a logical processor entry to an isolated enabled state, and a logical processor withdrawal from an isolated enabled state.

The configuration storage 250 stores configuration parameters to configure an access transaction generated by the processor 110. The processor 110 has a normal execution mode and an isolated execution mode. The access transaction includes access information 226 as discussed above. The configuration storage 250 receives the operand 224 from the instruction decoder and execution unit 210 (Figure 2A) and includes a processor control register 252, an isolated setting register 260, and an isolated subsystem memory range setting register 280. The processor control register 252 contains an execution mode word 253. The execution mode word 253 is asserted as an execution mode word signal when the processor 110 is configured in the isolated execution mode. In one embodiment, the execution mode word 253 stores the ID of the currently active subsystem operating in the isolated execution mode or an indicator that the processor is not currently in isolated execution mode.

The isolated setting register 260 contains an isolated setting 268. The isolated setting 268 defines the isolated memory area (e.g. the isolated area 70 in the accessible physical memory 60 shown in Figures 1B and 1C). The isolated setting register 260 includes a mask register 262, a base register 264, and a length register 266. The mask register 262 stores an isolated mask value 263. The base register 264 stores an isolated base value 265. The length register 266 stores an isolated length value 267. The isolated mask, base, and length values 263, 265, and 267 form the isolated setting 268 and are used to define the isolated memory area. The isolated memory area may be defined by using any combination of the mask, base, and length values 263, 265 and 267. For example, the base value 265 corresponds to the starting address of the isolated memory area, while the sum of the base value 265 and the length value 267 corresponds to the ending address of the isolated memory area.

The isolated subsystem memory range setting register 280 stores the isolated subsystem memory range settings 281. The isolated subsystem memory range settings 281 can be represented as a table having a plurality of rows and columns that store the settings for the subsystems within the isolated area. This table may represent the settings for all the possible subsystems in the isolated memory area, or it may represent a cache of settings for selected subsystems. Each row corresponds to a different subsystem and has a number of column entries that describe the subsystem and the subsystem's associated memory zone (75_0-75_N) in the isolated memory area 70 (Figure 1C). Each row includes an ID value 283, an entry attribute 285, a subsystem range setting 287, and an entry context 288. The ID value is a unique identifier for each subsystem. The entry attribute 285 includes a plurality of values including page size (such as large or normal sized), read or write access, and a master subsystem flag.

Each subsystem identified by the isolated subsystem memory range settings 281 includes a subsystem range setting 287 having a subsystem mask value, a subsystem base value, and a subsystem length value, as will be discussed in more detail, that defines a memory zone (75_0-75_N) for a respective subsystem in the isolated memory area 70 (Figure 1C). Each subsystem identified by the isolated subsystem memory range settings 281 also includes an entry context 288. The entry context 288 is used for entry by a subsystem into its associated memory zone for isolated execution.

One subsystem is defined as the master subsystem, and given a pre-defined fixed ID 283. This master subsystem is configured to include the entire isolated memory area. The processor nub is loaded into one portion of the master subsystem memory. All other subsystems are configured as subsets of the master subsystem, and exclude the memory used by the processor nub. The code executing in the master subsystem provides the management and maintenance functions for the other subsystems. Only code executing from the master subsystem may change the content of the isolated subsystem memory range setting register 280, thereby changing the configuration of the other subsystems.

When a master subsystem is initialized it fills in a row of the isolated subsystem memory range setting register 280 for itself including: setting its predetermined ID value 283, setting the master subsystem flag as a fixed entry in the entry attributes 285 to identify itself as the master subsystem, defining its associated memory zone using the subsystem range setting 287, and setting its entry address in the entry context 288. The isolated subsystem memory range setting register 280 is controlled by the master subsystem executive which corresponds to the processor nub 18 (or processor executive (PE)) of the master subsystem 71 (Figure 1C). As the master subsystem initializes new subsystems it creates new rows in the isolated subsystem memory range setting register 280 defining each new subsystem.

To pass control from a first subystem to a second subystem, a subystem control change instruction is issued. The subsystem control change instruction uses the second subsystem ID which is called as a parameter. The processor checks the IDs 283 of the isolated subsystem memory range settings 281 for a match, and if a match is found, the processor extracts the subsystem range setting 287 (e.g. subsystem mask value, subsystem base value, and subsystem length value to define the memory zone of the second subsystem) and the entry context field 288 of the second subsystem and copies them into the current subsystem range setting configuration register 273 and the current entry context configuration register 275, respectively. These new configuration registers represent the configuration for the currently active subsystem, and are used to drive the multi-memory zone access checking circuit 270, as will be discussed.

The current subsystem range setting configuration register 273 includes a subsystem mask register 289, a subsystem base register 291, and a subsystem length register 296. The subsystem mask register 289 stores a subsystem mask value 290. The subsystem base register 291 stores a subsystem base value 292. The subsystem length register 296 stores a subsystem length value 297. The memory zone (75_0-75_N) of the isolated memory area 70 (Figure 1C) for the currently active subsystem may be defined by any combination of the subsystem mask, base, and length values 290, 292, and 297. For example, the base value 292 corresponds to the starting address of the memory zone for the currently active subsystem, while the sum of the base value 292 and the length value 297 corresponds to the ending address of the memory zone for the currently active subsystem. The current entry context configuration register 275 includes an entry frame register 294 to store an entry frame address 295 copied from the entry context field 288. The entry address 295 is the physical address of the entry frame.

The processor can then enter into an isolated execution mode (e.g. iso_enter) under the control of the second subsystem. To do this, the processor loads an entry frame into a frame register set. The entry frame is pointed to by the entry address 295 in the entry frame register 296. The processor is now configured in an isolated execution mode under the control of the second subsystem, which is now the currently active subsystem, the memory zone of which is identified by the values stored in the current subsystem range setting configuration register 273, as previously discussed.

However, if an ID match does not occur, and a corresponding row of the isolated subsystem memory range settings 281 does not exist for the second subsystem, the processor checks the isolated subsystem memory range settings 281 for the master subsystem, which is identified by the set master subsystem flag in the entry attributes 285. The processor loads the parameters of the corresponding row of the subsystem isolated subsystem memory range settings 281 for the master subsystem and passes control to the master subsystem. Upon entry of the master subsystem, the processor determines that the second subsystem parameter ID is not for the master subystem, but a different subsystem. If the second subsystem exists, the master subsystem defines the second subsystem in the isolated subsystem memory range settings 281 by creating a new row for the second subsystem. If the memory range setting register 280 is a cache, with fewer entries than the possible subsystems, and there are no unused entries available, then the master subsystem may evict one of the existing entries to replace it with the new row for the second subsystem. Only the master subsystem entry cannot be replaced. Control is then passed from the master subystem (i.e. the first subsystem) to the second subystem as previously discussed. However, if the second subystem does not exist, the processor generates a failure or fault condition.

The multi-memory zone access checking circuit 270 checks the access transaction using at least one of the configuration parameters (e.g. the execution mode word 253 and the current subsystem range setting configuration 273) and the access information 226. The multi-memory zone access checking circuit 270 generates the processor isolated access signal 272 and the access grant signal 274. The processor isolated access signal 272 is asserted when the processor 110 attempts a memory access to a physical address within the isolated memory area 70. The access grant signal 274 indicates whether a memory access request can be granted. The access grant signal 274 is asserted if the requested memory address is outside the isolated memory range, OR, the processor is currently executing in isolated mode AND the address is within the currently active isolated subsystem memory range.

Figure 3 is a diagram illustrating the multi-memory zone access checking circuit 270 to manage accesses for isolated execution according to one embodiment of the invention. The multi-memory zone access checking circuit 270 includes an isolated area address detector 310, a subsystem address detector 311 and an access grant generator 315.

The isolated area address detector 310 receives the isolated setting 268 (e.g., the isolated mask value 263, the isolated base value 265, the isolated length value 267) from the configuration storage 250 in Figure 2B. The isolated area address detector 310 detects if the physical address 282 is within the isolated memory area defined by the isolated setting 268. The isolated area address detector 310 includes a masking element 312, a comparator 314, and a combiner 318. The masking element 312 masks the physical address 282 with the isolated mask value 263. In one embodiment, the masking element 312 performs a logical AND operation. The comparator 314 compares the result of the masking operation done by the masking element 312 and the isolated base value 265, and generates a comparison result. In other embodiments, the comparator 314 can also use the isolated length value 267 in generating the comparison result. The combiner 318 combines the comparison result with other conditions to generate the processor isolated access signal 272. The processor isolated access signal 272 is asserted when the physical address 282 is within the isolated memory area as defined by the isolated mask and base values 263 and 265, and in some embodiments, the length value 267.

The subsystem address detector 311 receives the values of the current subsystem range setting configuration register 273 (e.g., the isolated subsystem mask value 290, the isolated subsystem base value 292, and the isolated subsystem length value 297) from the configuration storage 250 in Figure 2B and receives access information 226 (e.g. a physical address 282). The subsystem address detector 311 detects if the physical address 282 is within a subsystem's associated memory zone as defined by the values of the current subsystem range setting configuration 273. The subsystem address detector 311 includes a masking element 322, a comparator 324, and a combiner 328. The masking element 322 masks the physical address 282 with the subsystem mask value 290. In one embodiment, the masking element 322 performs a logical AND operation. The comparator 324 compares the result of the masking operation done by the masking element 322 and the subsystem base value 292, and generates a comparison result. In other embodiments, the comparator 324 can also use the isolated length value 297 in generating the comparison result. The combiner 328 combines the comparison result with other conditions to generate a subsystem address matching signal 334. The subsystem address matching signal 334 is asserted when the physical address 282 is within a currently active subsystem's associated memory zone (75_0-75_N) as defined by the values of the current subsystem range setting configuration 273.

The access grant generator 315 combines the subsystem address matching signal 334, the execution mode word 253, and the processor isolated access signal 272 to generate the access grant signal 274. The access grant signal 274 indicates whether a memory access request can be granted. The access grant signal 274 is asserted and a memory access is granted, if the requested memory address is outside the isolated memory area 70, OR, the processor is currently executing in isolated mode AND the address is within the currently active isolated subsystem memory range. For example, when the processor isolated access signal 272 is not asserted, the requested physical address is outside the isolated memory range and the access grant signal 274 is asserted to grant a memory access to the non-isolated memory area. Alternatively, the access grant signal

274 is asserted when the processor isolated access signal 272 is asserted and both the subsystem address matching signal 334 and the execution mode word 253 for the currently active subsystem are asserted. This indicates that an isolated access to the memory zone in the isolated memory area for the currently active subsystem is valid or allowed as configured. In one embodiment, the access grant generator 315 performs a logical AND operation of these signals. Thus, only if the physical address requested by a processor, operating in an isolated execution mode, is within the correct memory zone for the currently active subsystem will the access transaction be granted.

Figure 4 is a flowchart illustrating a process 400 to control access to multi-memory zones of the isolated memory area for isolated execution according to one embodiment of the invention.

Upon START, the process 400 defines as an isolated memory area using the isolated setting (e.g., isolated mask value, isolated base value, and length value) (block 410). Next, the process 400 defines isolated memory zones for each subsystem in the isolated memory area using isolated subsystem range settings (e.g., subsystem mask and base values, ID's, entry attributes, and entry contexts) (block 415). Then, the process 400 receives access information from an access transaction. The access information includes a physical address as provided by the Translation Lookaside Buffer and access type (block 420). The process 400 then determines if the physical address is within the isolated memory area (block 425). If not, then the process 400 generates an access grant signal for normal execution (block 430).

If the physical address does fall within the isolated area of memory, then the process 400 determines whether the physical address falls within the memory zone of the isolated memory area for the currently active subsystem (block 435). If not, the process 400 generates a failure or fault condition (block 440). However, if the physical address does fall within the memory zone for the currently active subsystem, then the process 400 determines if the execution mode word signal is asserted for the currently active subsystem (block 445). If not, the process 400 generates a failure or fault condition (block 450). Otherwise, the process generates an access grant signal to the memory zone for the currently active subsystem for isolated execution (block 455). The process 400 is then terminated. Thus, only if the physical address requested by a processor, operating in an isolated execution mode, is within the correct memory zone for the currently active subsystem will the access transaction be granted.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.